LAWRENCE
LIVERMORE
NATIONAL
LABORATORY

UCRL-CONF-228878

Intelligent Classification and Visualization of Network Scans

L. Chen, C. Muelder, K. Ma, A. Bartoletti

March 9, 2007

ACM SIGKDD 2007
San Jose, CA, United States
August 12, 2007 through August 15, 2007

This work was performed under the auspices of the U. S. Department of Energy by University of California, Lawrence Livermore National Laboratory under Contract W-7405-Eng-48.

Intelligent Classification and Visualization of Network Scans

Lei Chen Chris Muelder Kwan-Liu Ma
University of California, Davis
Tony Bartoletti
Lawrence Livermore National Laboratory

ABSTRACT

Network scans are a common first step in a network intrusion attempt. In order to gain information about a potential network intrusion, it is beneficial to analyze these network scans. Statistical methods such as wavelet scalogram analysis have been used along with visualization techniques in previous methods. However, applying these statistical methods to reduce the data causes a substantial amount of data loss. This paper presents a study of using associative memory learning techniques to directly compare network scans in order to create a classification which can be used by itself or in conjunction with existing visualization techniques to better characterize the sources of these scans. This produces an integrated system of visual and intelligent analysis which is applicable to real world data.

Categories and Subject Descriptors

I.2.0 [**Artificial Intelligence**]: General; I.3.8 [**Computer Graphics**]: Applications; I.5.2 [**Pattern Recognition**]: Design Methodology—*pattern analysis*

General Terms

Design, Performance, Security

Keywords

Associative Memory, Classification, Computer Security, Information Visualization, Machine Learning, Network Scans, Pattern Reconstruction

1. INTRODUCTION

This paper presents an intelligent system approach to visual characterization of network scans. This characterization process is a useful tool for analysts in counterintelligence efforts against potential network intruders. Scanning a network is a common first step in a network intrusion attempt. The process of scanning a network is usually performed to determine what exists on a network. For example, if an attacker is looking for exploitable web servers, then he or she would attempt to connect on TCP/UDP port 80 to every possible IP address within a certain range. If there is a web server using port 80 at any of these IP address, it will probably respond. However, for addresses where there is nothing, or where there is a computer that is not running a web server, there will be no response. Detecting these scans is fairly easy, but mining them for information about the attacker can be relatively difficult.

An attacker can do several things in an attempt to make such a scan anonymous; for instance, coming from different source addresses or scanning destination addresses in a random order. In fact, it is even possible to perform a scan indirectly by using a fake source address so the scan looks like it is coming from a different computer. Denial of service and worm propagation attacks can also produce scan-like behavior, and since they do not need the target to respond, they often fake their source addresses as well. The port number is also not sufficient for categorizing scans, because both malicious and benign scans can often be run on the same port number. For example, both a web crawler and a worm that targets webservers would target port 80. Therefore, some other metric must be used for categorization purposes.

Experimental results have shown that variations in arrival time of the scanning connections often have a high correlation with particular sources [17]. That is, the timing information produces a digital 'fingerprint' that correlates to a particular source. It is surmised that this correlation is due to a combination of factors, including the connection application software employed, the supporting hardware platform, operating system characteristics, and regular interference from other processes on the source system that compete for these resources. Network location based factors such as number of hops and nature of intervening routers are certainly responsible for some degree of the timing structure as well, and make this a particularly interesting and challenging problem in network traffic forensics.

The critical question analysts seek to answer is whether the same source ensemble run in an entirely different network (hence, different source IP address), would exhibit a timing structure that is sufficiently similar. This would uniquely identify the environment, and, by extension, the actor behind the observed activity. Alternatively, the analysts would like to know the degree to which the effects of intervening routers produce characteristic packet timing irregularities for different activities conducted from a constant network

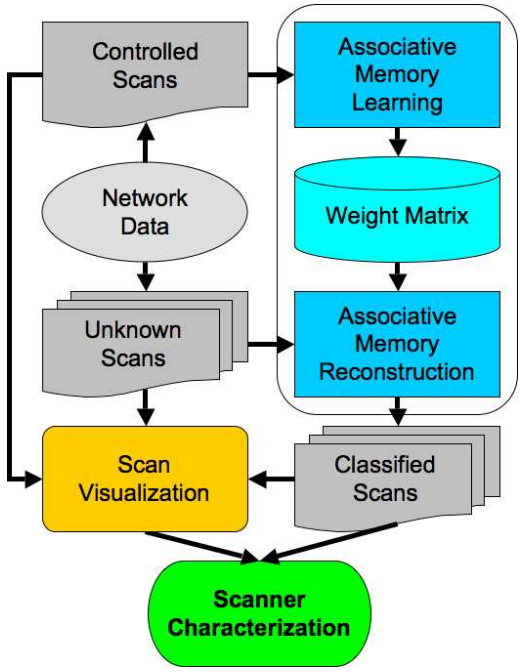


Figure 1: *The overall methodology.* Known and unknown scans are collected off the network. The known scans are used to train the associative memory in order to create a weight matrix which can then be used to classify the unknown network scans. All of these scans can be visualized, with the overall goal of gaining information about the sources of the scans.

location. This latter capability would provide a means to determine the veracity of a given source IP address when faced with potential address-spoofing.

To answer this question, analysts must first be able to correlate scans from different source addresses based on timing information. To accomplish this, the analysts need to compare the timing information of very large quantities of network scans quickly and efficiently. Previous methods have used statistical reduction and visualization to compare these scans. However, the statistical reduction has an inherent data loss and can be susceptible to noise problems; and the direct visualization techniques can become quite unwieldy as the number of scans increases. Intelligent pattern recognition algorithms are quite good at dealing with these issues. They are good at reconstructing distorted or incomplete data, and they scale well to large numbers of inputs. So, an artificial intelligence based classification methodology was developed that can be used both alone and in conjunction with existing visualization techniques to better characterize potentially hostile scan sources.

1.1 Methodology

The methods used in this paper cover a wide variety of techniques, including statistical analysis, visualization, and artificial intelligence, with the ultimate goal of aiding in counterintelligence efforts by characterizing potential adversaries through their scanning activity. An overview of these techniques is shown in Figure 1. In order to analyze and characterize network scans, they must first be detected and extracted from the raw network data. This is currently being done though the use of some fairly simple statistical data mining techniques. These techniques consist of statistically detecting a scan in progress, then extrapolating backwards and forwards in time in order to find when the scan starts and stops. Most of the scans that are detected and extracted in this manner are caused by unknown sources, and so they form a set of unknown scans. But some of the extracted scans originated from known sources, where the parameters of the scans were carefully controlled. That is, properties such as the source hardware, software, scanning tool, and location on the internet are known. This creates a set of controlled scans, which can then be used as a ground truth to compare against the unknown scans. Due to the constant scanning activity occurring all the time on the internet, these sets of scans can rapidly grow to be quite large, so it is infeasible to compare them against each other by hand. The visualization techniques of [16] are one set of methods being used to deal with the scalability issues through statistical analysis and visual presentation of both the controlled and unknown scans. These scans can also be used as inputs to an artificial intelligence algorithm, which is the focus of the approach presented here. The controlled scans are used as training data for an associative memory learning process which generates a weight matrix. This weight matrix can then be used in an associative memory reconstruction process to take the unknown scans and classify them. These classified scans can then either be used directly to characterize their sources, or they can be used in the visualization system to improve its effectiveness at scanner characterization.

1.2 Previous Work

The study of network security has been popular for the last decade. Visualization systems have been developed to visualize and compare the network scan pattern in order to detect the potential for attacks. Muelder et al. [16] present a means of facilitating the process of characterization by using visual and statistical techniques to analyze the patterns found in the timing of network scans. The system allows large numbers of network scans to be rapidly compared and subsequently identified. Conti et al. [1] use a parallel coordinates system to display scan details and characterize attacks. Other network visualization tools that show scans exist, such as PortVis [15], but these tools generally focus on the detection of suspicious activity and not on the analysis of such activity. Also, all these systems present the original scan data, which contain noise and distortion.

Machine learning methods - associative memory models in particular - have been widely applied in the pattern recognition and classification area. Tavan et al. [21] extend the neural concepts of topological feature maps towards self-organization of auto-associative memory and hierarchical pattern classification in 1990. Stafylopatis et al. [20] proposed a technique based on the use of a neural network model for performing information retrieval in a pictorial information system. The neural network provides auto-associative memory operation and allows the retrieval or stored symbolic images using erroneous or incomplete information as input. In Y. Dai et al's paper[5] [4], an associate

memory model is proposed which utilizes the facial action feature rate of occurrence on happiness, easiness, uneasiness, disgust, suffering, and surprise.

Machine learning algorithms have also been directly applied to computer security problems in the past. [6] uses a machine learning technique to analyze log files and look for anomalies. These techniques have also been applied to intrusion detection systems in several approaches [11, 18, 19]. However, all these approaches focus on using the machine learning to aid in the detection of malicious activity, rather than the analysis of it, as is done in this paper.

2. SCAN DATA AND VISUALIZATION

The scan data used in this paper was generated and collected by the Computer Incident Advisory Capability (CIAC) group from the network at Lawrence Livermore National Lab. The visualization techniques used are introduced in [16].

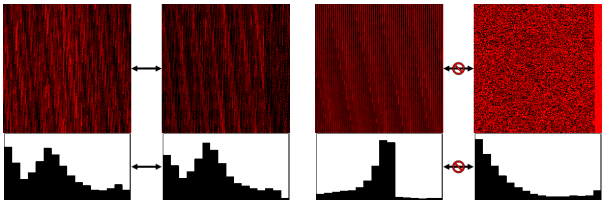
2.1 Network Scan Data

Each scan consists of timing information of a scan over a class B network, which contains 65,536 destination addresses. The scan data in its most raw form consists of pairs of destination addresses and times, one for each probe in the scan. In this form, the data is not very easy to work with directly, so various transformations were performed to create a set of modes to visualize different aspects of the data. For example, mode 20 is used to visualize the number of connections per unique address and mode 21 is to visualize the time span between the first connection attempt and the last connection attempt to each address. Some selected modes are listed below:

- *mode-20*: f(a) = N(v), number of visits per unique address
- *mode-21*: f(a) = tFirst - tLast, the revisit-span for each address
- *mode-22*: f(a) = tFirst - E(tFirst), time deviance for first probes
- *mode-23*: f(a) = tLast - E(tLast), time deviance for last probes
- *mode-24*: f(a) = d(tFirst), time delta on sequential addresses, first probe
- *mode-25*: f(a) = d(tLast), time delta on sequence

2.2 Wavelet Analysis

While each of these data modes transform the scans into a regular form, a direct comparison would not yield useful results. For instance, if an attacker scanned every other network address one day, then came back the next day and scanned the addresses that were skipped, then a direct comparison would reveal no similarity, even though the patterns would be nearly identical; they would just be out of phase from each other. However, there are several algorithms utilizing frequency analysis that are useful for handling this kind of data, such as Fourier transforms and wavelet analysis. Although network scan patterns can exhibit periodic or quasi-periodic structure, they often contain gaps, aperiodic



(a) Similar patterns (b) Dissimilar patterns

Figure 2: Applying wavelet scalograms creates a representation that is directly comparable, at the cost of data loss.

aberrations, and regions where the relative phase of the periodic structures has shifted. These are things that Fourier analysis has been found to handle poorly [7].

So wavelets are used because they are relatively resistant to phase shifts and noise. This means that similar patterns will have similar wavelet scalograms, even if the patterns are shifted slightly or different parts of the pattern are missing, as can be seen in Figure 2(a). But dissimilar patterns will still produce different scalograms, as can be seen in Figure 2(b). There are several variations on the wavelets used, the simplest of which is as follows. Given a series of $N = 2^n$ items $D_0 = (d_{0,1}, d_{0,2}, ..., d_{0,N})$, we can calculate recursively:

- $D_k = (d_{k,1}, d_{k,2}, ...d_{k,2^{n-k}}) = (\frac{d_{k-1,1}+d_{k-1,2}}{2}, ..., \frac{d_{k-1,2^{n-k}-1}+d_{k-1,2^{n-k}}}{2})$
- $S_k = (s_{k,1}, s_{k,2}, ...s_{k,2^{n-k}}) = (\frac{|d_{k-1,1}-d_{k-1,2}|}{2}, ..., \frac{|d_{k-1,2^{n-k}-1}-d_{k-1,2^{n-k}}|}{2})$
- $\sigma_k = \sum \frac{S_k}{2^{n-k}}$

for $0 < k < n$. At each recursion the σ values are the mean of the corresponding data series, which estimates the variance at each resolution. More complicated wavelets can be calculated by changing the functions used to calculate D_k and S_k, and are described in [16]. Once these wavelet scalograms are calculated, they can be directly compared to one another. The downside to using these wavelets is that they lose fine details in the data, such as where a particular feature occurs in the pattern. Still, they can be used to create a good approximate overview from which other information can be shown.

2.3 Visual Representation

The visual representations of these scans use essentially the same techniques that have been presented in [16]. Namely, individual scans are presented in detail in a 256x256 grid, where the x and y axes represent the third and fourth bytes of the address respectively. Two examples of scans presented in this way are shown in Figure 3. Also, in order to represent large numbers of these scans at once, a graph representation of the scans was used. In this view, each scan is a node in a complete graph, where edges are weighted by how similar the scans are according to the wavelet analysis. A force-directed layout is then applied to this graph, and clusters of

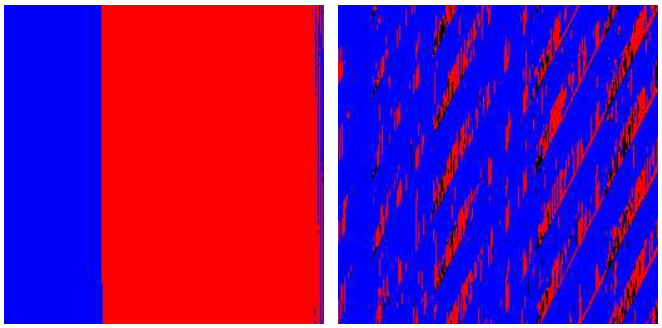


Figure 3: Sample network scan data patterns of mode 22

similar scans are grouped together. Edges that are below a threshold are dropped for clarity. This creates an overview in which large scale trends, such as clusters, can be seen. It also provides an interface for selecting and viewing scans of interest in more detail. An example of such a graph is shown in Figure 4.

2.4 Binary Coding

For the purposes of this paper, data mode 22 is used and the data is binary encoded. This is necessary because the bidirectional associative memory algorithm only works on binary patterns. In order to extend this algorithm to non-binary patterns, more than one bit must be used per element in the pattern. In the simplest encoding used in this paper, two bits are used for each IP address in the scan. If the captured time of the first probe is earlier than the expected time, we give the neuron value "11;" if it is later than the expected time we give the value "00;" otherwise it is assigned "10." Figure 3 gives an example of two network scan patterns of mode 22. The blue pixels are neurons with value "11," red pixels are neurons with value "00," and black pixels are neurons with value "10." More bits can be used to break down the continuous range of possible values into a finer set of discrete ranges, but this would increase the computational cost and memory usage. The system has two stages: encoding stage and decoding stage. In the encoding stage, we input all the controlled patterns and encode them in one weight matrix. In the decoding stage, we input the deviant patterns and the system will return the reconstructed patterns using the weight matrix we created in the encoding stage.

3. APPROACH

There are two main categories of machine learning algorithms: unsupervised and supervised. An unsupervised algorithm approaches a data set with little to no pre-conceived notions about the classes of items that it contains; it attempts to separate the items into statistically distinct groups on its own. Examples of unsupervised algorithms include k-means clustering and self- organizing maps [13, 10]. But, unsupervised algorithms cannot effectively use controlled data. Since there is controlled scan data available, a supervised algorithm is more readily applicable to the task presented here. A supervised algorithm is given a large number of examples of items in each class it is to detect; each is labeled with the class to which it belongs. Then, when new items are presented, it puts them into the class they are most likely to belong to. Examples of supervised algorithms include neural networks [14, 22], support vector machines [2, 3], and associative memory [9]. Of these, associative memory in particular has been shown to be useful for pattern recognition, and so it was selected for use here.

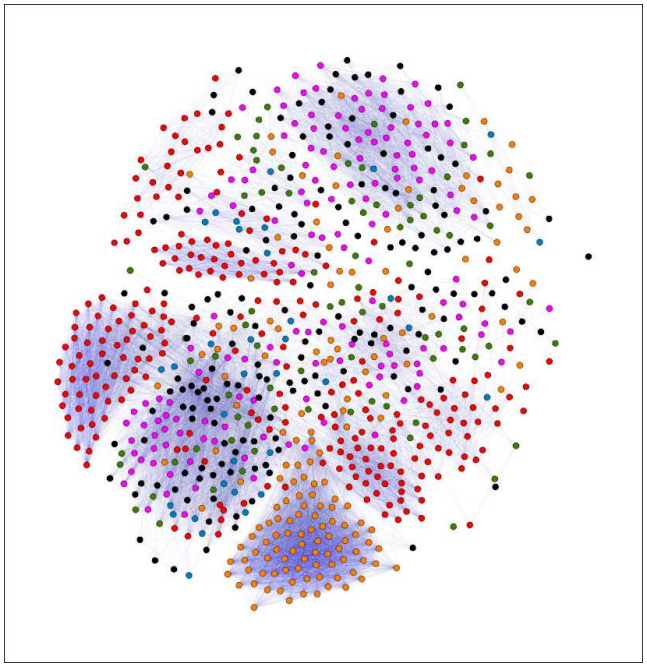


Figure 4: A graph of network scans generated through statistical analysis using wavelet scalograms. Each node represents a scan, and each edge represents a measure of how similar the scans it connects are. The scans are colored according to port number.

3.1 Associative Memory Method

The concept of associative memory was first proposed by Kohonen T. in [9]. In the human mind, the memory process takes a stimulus and matches it up with what it remembers about that stimulus. The nature of associative memory is an emulation of human memory, and so it works in a similar way. The associative memory model has many applications, for example, pattern cognition and reconstruction, image processing, face, character and voice recognition, databases, control systems, and robotics. Human memory is quite robust in that in can correct errors and recognize stimuli even when the they are incomplete or distorted. Similarly, in associative memory, given a stimulus pattern that is distorted or incomplete, associative memories are often able to reproduce correct response pattern. It does this by remembering a set of known patterns and then matching up incoming unknown patterns against them.

There are a variety of types of associative memory models that have been studied in the last two decades, including Hopfield Networks [8] and Bidirectional Associative Memory (BAM) [12]. While initial results were produced with Hopfield Networks, it was found that BAM was much faster and produced results that were just as good, so BAM was

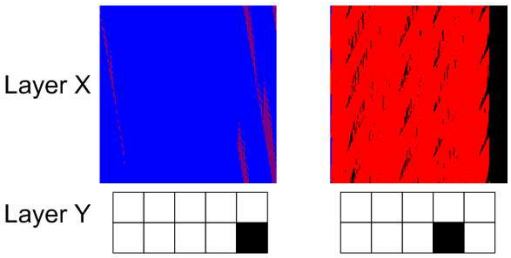


Figure 5: The sample pairs of patterns on layer X and layer Y .

used for this paper.

3.2 The BAM algorithm

The BAM is one of the hetero-associative memories, such that the X layer and Y layer of the network have distinct dimensions. The BAM structure is commonly built based on a neural network. The BAM model maps stimulus vectors to response vectors $\{(X_1, Y_1), \ldots, (X_i, Y_i), \ldots, (X_N, Y_N)\}$. We use bipolar mode for the two states of the neurons: fire is 1 and not fire is -1. Therefore, X_i is $\{1, -1\}^n$ and Y_i is $\{1, -1\}^m$. There is a weight between each pair of the neurons in layer X and layer Y. There is no weight between neurons within the same layer. It has two-way retrieval capabilities: X_i ¡-¿ Y_i In the layer X, each pattern has 65536 (256^2) pixels and we use two bits for each pixel. In total, there are 131,072 (65536 * 2) neurons. In layer Y, the values of the neurons are encoded using the index of the corresponding pattern in layer X. Here we use 10 neurons for layer Y, which allows for memory of up to 2^{10} control patterns. For example, in the first control pattern, the index is 1, so the value of the layer Y neurons will be (-1,-1,-1,-1,-1,-1,-1,-1,-1,1).

The two patterns in layer X and layer Y will be stored as a pair, some examples of this are provided in Figure 5 There is a weight between any pair of the neurons. For example, let us have x_i for the value of the i^{th} neuron in layer X and y_j for the value of the j^{th} neuron in layer Y of the k^{th} pair of controlled patterns. The weight matrix can be calculated by:

$$W_{ij} = \sum_{k=1}^{K} x_{ij}^k * y_{ij}^k \tag{1}$$

where k is the k^{th} pair of controlled patterns.

In the decoding stage, the system is given an unknown pattern, which maybe a distorted or incomplete controlled pattern. The system will decode the deviant pattern and output the original controlled pattern. When it goes through the network, each neuron in the network is updated by:

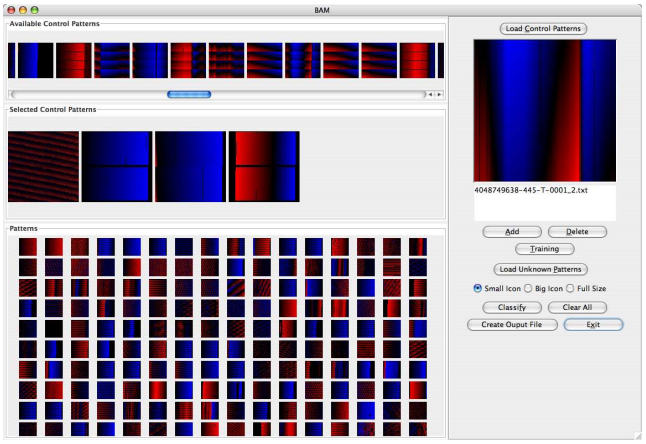


Figure 6: The user interface. The work flow runs from top to bottom. The top row shows a set of patterns from which control patterns can be selected. The middle row shows the set of contrl patterns that have been selected to train on. And the bottom section shows the set of unknown scans for the system to classify.

$$y = \begin{cases} 1 & \text{if } \sum_{j=1}^{131072} W_{ij} * x_j > \theta \\ \text{previous } y_i & \text{if } \sum_{j=1}^{131072} W_{ij} * x_j = \theta \\ -1 & \text{if } \sum_{j=1}^{131072} W_{ij} * x_j < \theta \end{cases} \tag{2}$$

$$x = \begin{cases} 1 & \text{if } \sum_{i=1}^{10} W_{ij} * y_i > \theta \\ \text{previous } x_i & \text{if } \sum_{i=1}^{10} W_{ij} * y_i = \theta \\ -1 & \text{if } \sum_{i=1}^{10} W_{ij} * y_i < \theta \end{cases} \tag{3}$$

where θ is a fixed threshold and n is the number of neurons. The threshold can be set by the user.

4. RESULTS

Application of the BAM algorithm to actual network scans yields some useful results. When applied to individual scans, it can be shown to be quite effective at removing distortions due to noise. When applied on a larger scale, it can effectively cluster a large set of scans quite rapidly. Finally, the clusters that it generates can reveal some interesting features in the data that statistical analysis missed. A picture of the interface is shown in Figure 6.

4.1 Classification Examples

To demonstrate how the BAM model works, let us use only four controlled patterns as the input. Figure 7 shows the visualization of the four patterns. The weight matrix is calculated using the input controlled patterns.

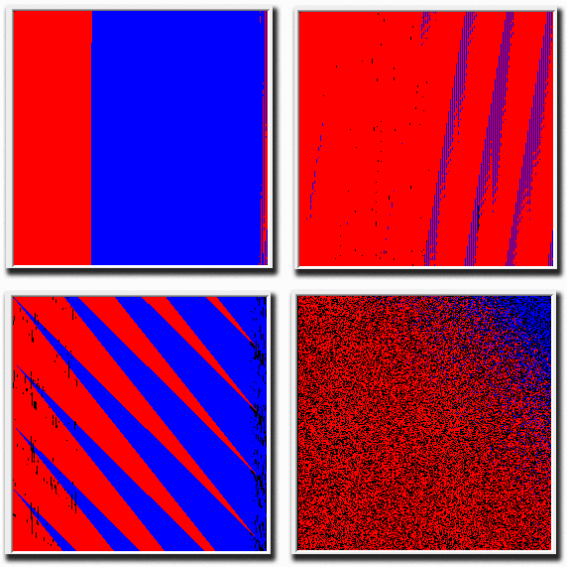


Figure 7: The four input patterns in our sample

Figure 8 shows the results of inputing one of the control patterns the system was trained on. As expected, the input scans at left are mapped exactly to the same scan in the output scans at right.

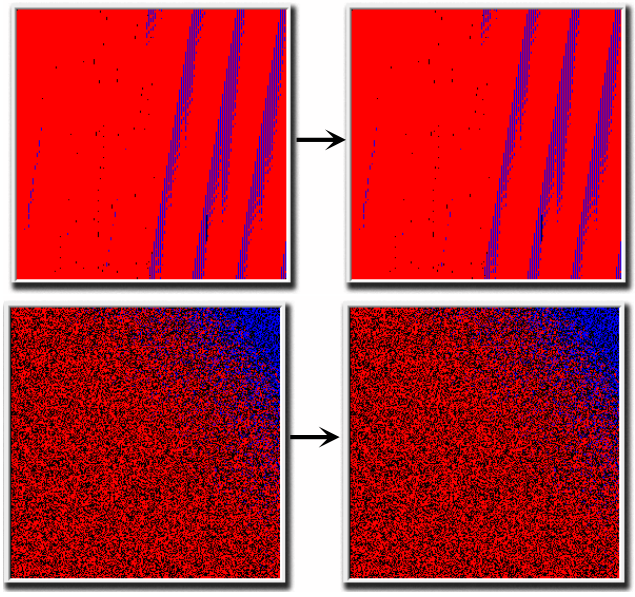


Figure 8: The system recalls the same pattern when we input one of the control patterns

In a second example, shown in Figure 9, the system was given a deviant pattern that nearly matched one of the control patterns, but contained noise or distortion. After reconstruction, the system recalled the original controlled pattern, and hence has matched up the distorted or incomplete pattern with the control data. The left images are the inputted scans and the right images are the control scans that the associative memory method matched them up with.

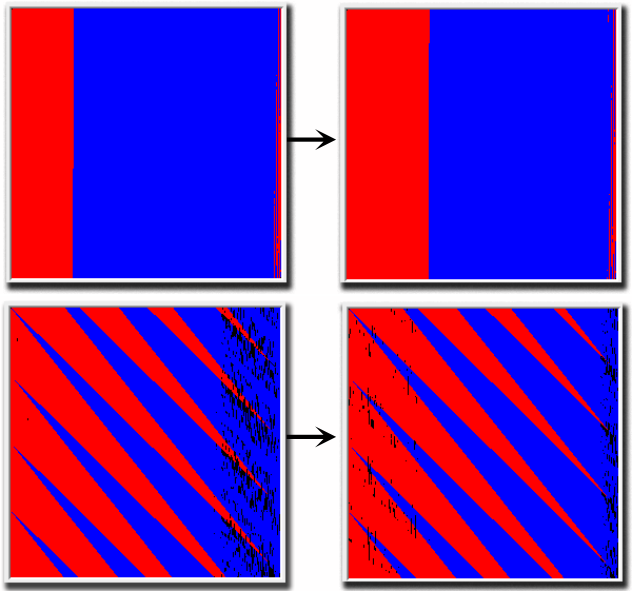


Figure 9: The system reconstructs the pattern when given a distorted pattern

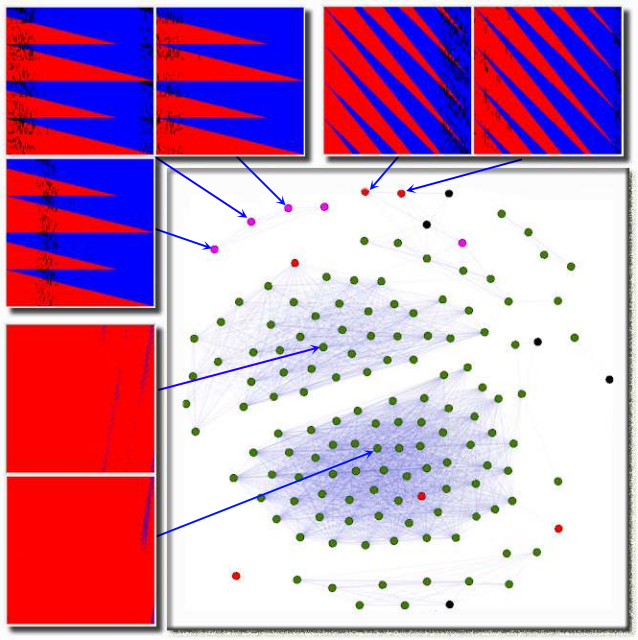


Figure 10: A graph showing the results of the BAM classification. Three main groups are colored, and examples from each are shown. Outliers are also shown as black nodes.

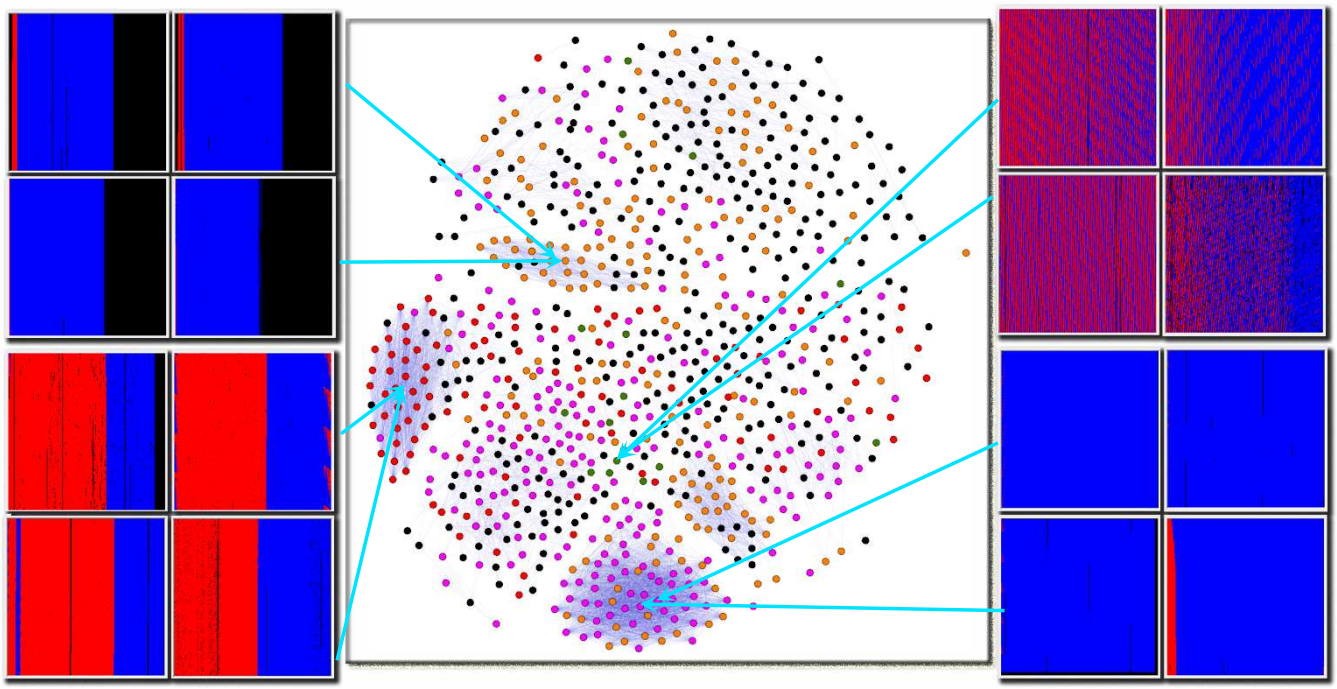


Figure 11: A graph showing the results of the BAM classification. Four main groups are colored, and examples from each are shown. Outliers are also shown as black nodes.

4.2 Visualization Integration

Once these unknown scan patterns have been classified by matching them up with their associated control patterns, it is useful to present these results visually to the user. This is particularly important as the number of classified scans increases and becomes too unwieldy to analyze by hand. For this work, it was decided that these results could best be visualized as a coloring of an existing graph generated by statistical means as in [16]. An example of this is shown in Figure 10, where the classification process was run on a small set of controlled scans. In this example, the set of scans was split into three groups, of which several samples are shown. In this image it can be seen that scans that have the same color do, in fact, have very similar patterns. It can also be seen that some scans did not match any of the controlled patterns. This is useful because it points out scans that are outliers, which should be viewed in more detail.

4.3 Case Studies

Figure 11 shows the results of applying the techniques presented here to color a graph of over 800 unknown scans collected off the network. One pattern that is readily apparent is the large number of unclassified scans, which are colored black in the image. This is indicative of the large number of different kinds of scans active on the internet that have not yet been identified. However, by applying an associative memory approach, these scans have been highlighted by a process of elimination, indicating a set of scans that would be beneficial to look at in more detail. Another prevalent pattern is that the nodes that were clustered by the statistical analysis are generally classified the same according to the associative memory. However, what is of interest is that some of the scans within a cluster are classified differently.

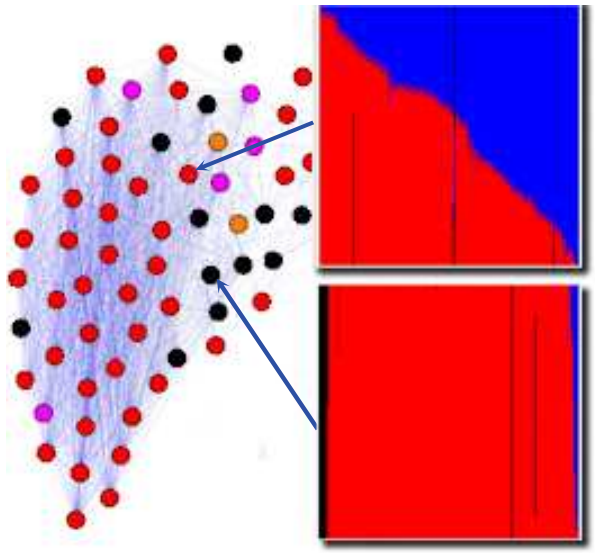


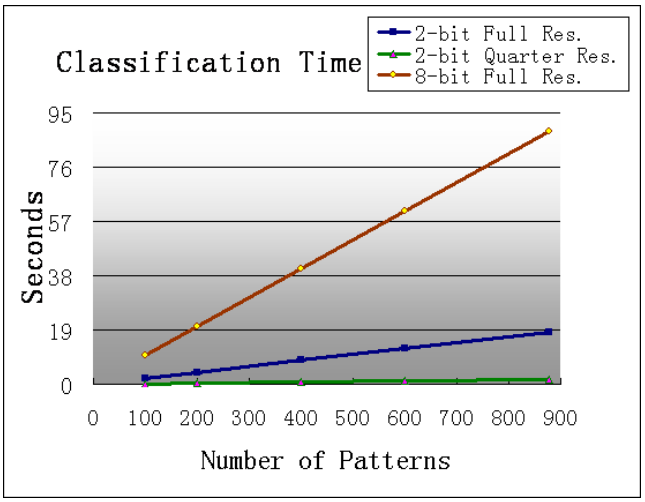
Figure 12: A close up of one cluster. Most nodes are colored red, and look as the detail view on top. But some are not, and can look different as is shown in the detail view at bottom.

This is indicating that the machine learning algorithm is picking up on a detailed pattern that is lost by this statistical analysis. As is shown in Figure 12, the difference in the patterns is not necessarily even difficult to discern by eye. This shows that machine learning techniques can quickly lead the analyst to find pattern details that were lost by the statistical analysis. Figure 14(c) shows what happens when the same test is run on a graph where the scans were represented with an 8 bit encoding instead (-127 to 127) of the 2 bit one. As can be seen, this completely changes the results, and provides another view of the data which can be investigated. Of interest here is that instead of the clusters being mostly classified by the BAM algorithm, they were marked as unknown. Also, the majority of the spread out scans that were not classified in Figure 11 are now classified.

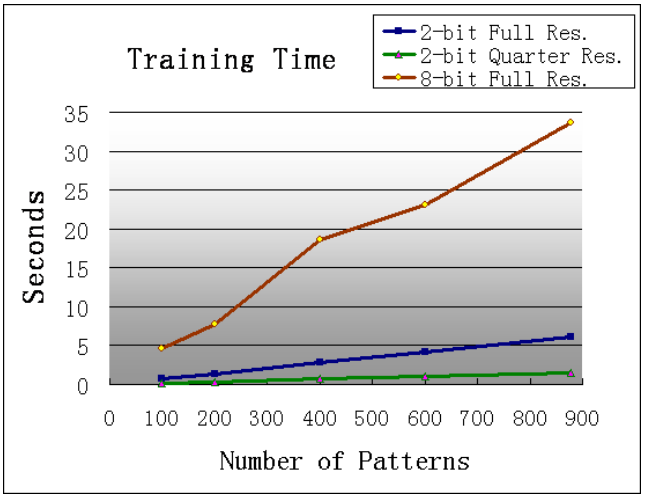
4.4 Performance Versus Quality

In order to accelerate the process, it was decided that it might be effective to only use a part of the pattern. So the process was run on samples of one fourth the size of the original data. Also, to get better quality results, it was decided that the process could be run with an 8 bit encoding of the data instead of the 2 bit encoding described earlier. Figure 15 shows what some of the patterns look like with 8 bit encoding. Some timing tests were run on a Mac Pro with a 2.66Ghz Dual-Core Intel Xeon, although only one core was used. The results of these tests are presented in the graphs in Figure 13, which show that. reducing the data boosts the speed of the process and increasing the bit encoding makes the process take longer.

In Figure 14, it can be seen that the results are mostly the same between the original graph and the lower quality graph, but the graph with 8 bit encoded patterns looks entirely different. In regions of Figure 14(b) that are fairly similar between two classes of scans, it is possible for scans to be misclassified by the approximation caused by undersampling. Specifically, in the purple cluster at the bottom of the graph, some of the nodes are supposed to be colored green, as can be seen in Figure 14(a). But since these nodes are clustered according to the statistical analysis, these scans actually are fairly similar and so this might actually be indicative of control patterns that were too similar, and hence perhaps should be grouped together into one class anyway. Figure 14(c) shows an entirely different coloring of the graph, where many nodes that were not classified in Figure 14(a) are now classified, and many nodes that were classified in Figure 14(a) are now not classified. This indicates that the quality of the results is greatly dependent on the number of bits used to represent the data. Since the cost was not increased substantially by the increase in the number of bits, the extra quality provided by this encoding is very affordable. Although most of the results presented here were done with only the two bit encoding, future results should probably be done with the eight bit or higher encoding.



(a) Training Time

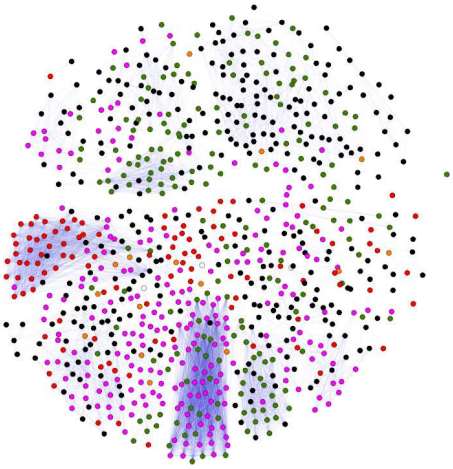


(b) Classification Time

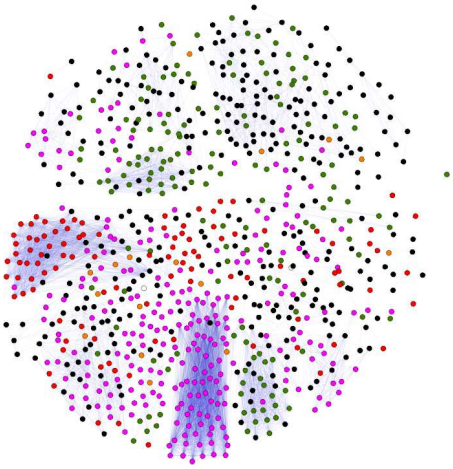
Figure 13: ***Performance boost through undersampling.*** **In both the training and classification stages, the process scales linearly by the number of patterns. The time required can be reduced by undersampling the patterns, at the cost of some quality of the classification. And the time requirements increase when the quality is improved by using more bits to encode the data.**

5. CONCLUSIONS

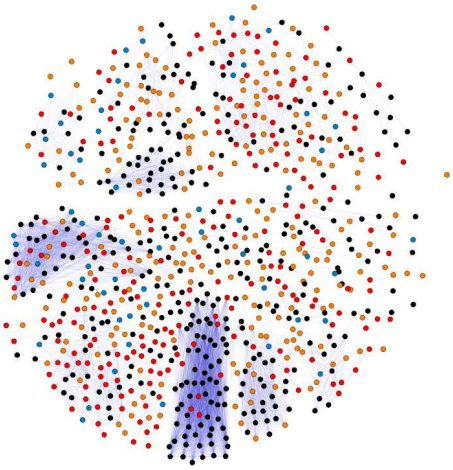
In the system presented in this paper, a machine learning method, associative memory, has been used to perform intelligent network scan pattern reconstruction and classification. When given a set of controlled scan patterns and a noisy or incomplete pattern, the results show that the sys-



(a) 2-bit Encoded, High Resolution



(b) 2-bit Encoded, Low Resolution



(c) 8-bit Encoded, High Resolution

Figure 14: Quality variation. (a) shows the original graph. (b) shows the results of using only $\frac{1}{4}$ of the data. (c) shows the results of using 8 bits to encode values instead of just 2. While (a) and (b) are mostly the same, there are some differences where (b) has lost some data, particularly in the large cluster at the bottom of the graph. (c) however is completely different from either of the other two, which shows that the results are very dependent on the bit encoding, so higher bit encoding should be done when possible.

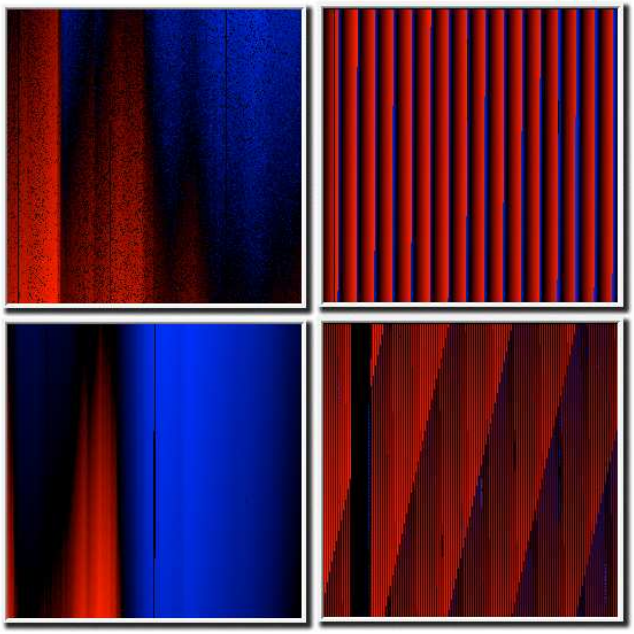


Figure 15: Some patterns encoded with 8 bits instead of 2. The quality is much greater and many more fine details can be seen.

tem can remove the noise and successfully return the complete pattern. These restored patterns are much more convenient for further studies, such as pattern comparison or pattern clustering, for the purpose of correlating malicious network activities. This paper has also shown the effectiveness of combining machine learning techniques with existing visualization and statistical techniques to create a useful visual representation for dealing with large numbers of network scans. Therefore, the results naturally lead to the feasibility of applying associative memory models in reconstruction and recognition of network scan patterns.

From an operational standpoint, one of the most important tasks in cyber security is "damage assessment". When damage (a successful intrusion, or a detected data-exfiltration activity) is discovered at one point, security managers must immediately seek evidence of this activity more broadly, where it may not yet have been discovered. Unfortunately, this assessment is typically limited to a search for the same (outsider) IP address, under the naive assumption that the hostile agent will be using the same IP address for their activities directed at range of targets. If the intruder or outside agent employs different IP address, either in time (evolving) or in space (intentionally to avoid naive correlation) then these broader damage assessments will fail.

A system by which an intruder's "pattern", such as exemplified by scan timing characterization, could be used to quickly "re-identify" an intruder (irrespective of IP address) would greatly enhance the effectiveness of damage assessment activities, enabling detection where none was previously possible. Our system demonstrates that associative memory techniques and visualization together form an effective basis for such a characterization system. The system could quickly be trained upon the known intruder evidence, and

then sought more broadly to determine the true scope of damage.

6. FUTURE WORK

One possible extension of this work would be to investigate looking at the differences between the original scans and the controlled scans that the associative memory classified them as. That is, by subtracting out the part of the pattern that is controlled, it should be possible to isolate the distortion patterns that are created by uncontrollable effects such as router delays. Doing so should greatly enhance the capability to correlate and identify such common but minute effects, even when the underlying scans are vastly different due to large scale effects such as the scanning tool used.

Another aspect that could be considered is the application of unsupervised algorithms to this task. These would produce a clustering without the need for controlled data. This could lead to an even more effective tool for finding outliers in the graph, since each true outlier would have a unique color instead of all being grouped into one class.

7. ADDITIONAL AUTHORS

8. REFERENCES

[1] G. Conti and K. Abdullah. Passive visual fingerprinting of network attack tools. *VizSEC/DMSEC '04: Proceedings of the 2004 ACM workshop on Visualization and data mining for computer security*, pages 45–54, 2004.

[2] C. Cortes and V. Vapnik. Support-vector networks. *Mach. Learn.*, 20(3):273–297, 1995.

[3] N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines and Other Kernel-Based Learning Methods*. Cambridge University Press, Cambridge, UK, 2000.

[4] Y. Dai, Y. Shibata, T. Ishii, K. Hashimoto, K. Katamachi, K. N. N. Kakizaki, and D. Cai. An associate memory model of facial expressions and its application: In facial expression recognition of patients on bed. *IEEE International Conference on Multimedia and Expo*, 2001.

[5] Y. Dai, Y. Shibata, Y. Nakano, and K. Hashimoto. Recognition of facial expressions based on the hopfield memory model. *Proceedings of IEEE ICMCS'99*, 12:133–137, 1999.

[6] L. Girardin and D. Brodbeck. A visual approach for monitoring logs. In *Proceedings of the 12th Usenix System Administration conference*, pages 299–308, 1998.

[7] A. Graps. An introduction to wavelets. *IEEE Computational Sciences and Engineering*, 2(2):50–61, 1995.

[8] J. Hopfield. Neural networks and physical systems with emergent collective computational abilities. In *Proceedings of the National Academy of Sciences*, volume 79. Washington, DC: National Academy Press, 1982.

[9] T. Kohonen. *Associative Memories: A System Theoretic Approach*. Spring-Verlag, Berlin, 1978.

[10] T. Kohonen. *Self-Organization and Associative Memory*. Springer-Verlag, Berlin, 3rd edition, 1989.

[11] A. Komlodi, J. Goodall, and W. Lutters. An information visualization framework for intrusion detection, 2004.

[12] B. Kosko. Bidirectional associative memories. *IEEE Trans. Syst. Man Cybern.*, 18(1):49–60, 1988.

[13] J. B. MacQueen. Some methods for classification and analysis of multivariate observations. In *Symposium on Math, Statistics, and Probability*, volume 1, pages 281–297, 1967.

[14] J. L. McClelland, D. E. Rumelhart, and the PDP Research Group. *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*. MIT Press, Cambridge, Massachusets, 1986.

[15] J. McPherson, K.-L. Ma, P. Krystosk, T. Bartoletti, and M. Christensen. Portvis: A tool for port-based detection of security events. *In ACM VizSEC 2004 Workshop*, pages 73–81, 2004.

[16] C. Muelder, K.-L. Ma, and T. Bartoletti. A visualization methodology for characterization of network scans. *Visualization for Computer Security, IEEE Workshops on*, pages 4–4, October 2005.

[17] B. Parno and T. Bartoletti. Internet ballistics: Retrieving forensic data from network scans. Poster Presentation, the 13th USENIX Security Symposium, Aug. 2004.

[18] L. Portnoy, E. Eskin, and S. Stolfo. Intrusion detection with unlabeled data using clustering, 2001.

[19] C. Sinclair, L. Pierce, and S. Matzner. An application of machine learning to network intrusion detection. In *ACSAC '99: Proceedings of the 15th Annual Computer Security Applications Conference*, page 371, Washington, DC, USA, 1999. IEEE Computer Society.

[20] A. Stafylopatis and A. Likas. A pictorial information retrieval using the random neural network. *IEEE Transactions on Software Engineering*, 18(7):590–600, July 1992.

[21] P. Tavan, H. Grubmller, and H. Khnel. Self-organization of associative memory and pattern classification: recurrent signal processing on topological feature maps. *Biological Cybernetics*, 64(2):95–105, Dec 1990.

[22] P. J. Werbos. *Beyond Regression: New Tools for Regression and Analysis in the Behavioral Sciences*. PhD thesis, Harvard University, Division of Engineering and Applied Physics, 1974.